SOFTWARE SYSTEM FACILITATING THE REPLACEMENT OR INSERTION OF DEVICES IN A COMPUTER SYSTEM

Related Application

The subject matter of U.S. Patent Application entitled "Method for Facilitating the Replacement or Insertion of Devices in a Computer System," filed on October 1, 1997, Application No. _____, and having attorney Docket No. MNFRAME.027A, is related to this application.

Priority Claim

The benefit under 35 U.S.C. § 119(e) of U.S. Provisional Application No. 60/046310, filed May 13, 1997 and entitled "High Performance Network Server System Management Interface," is hereby claimed.

Appendix A

Appendix A, which forms a part of this disclosure, is a list of commonly owned copending U.S. patent applications. Each one of the applications listed in Appendix A is hereby incorporated herein in its entirety by reference thereto.

Copyright Rights

Background of the Invention

Field of the Invention

The present invention relates to user interfaces for computer systems. More particularly, the present invention relates to implementing a graphical user interface (GUI) to allow for easy and efficient management and maintenance of peripheral devices in a computer network.

Description of the Related Technology

As enterprise-class servers, which are central computers in a network that manage common data, become more powerful and more capable, they are also

becoming ever more sophisticated and complex. For many companies, these changes lead to concerns over server reliability and manageability, particularly in light of the increasingly critical role of server-based applications. While in the past many systems administrators were comfortable with all of the various components that made up a standards-based network server, today's generation of servers can appear as an incomprehensible, unmanageable black box. Without visibility into the underlying behavior of the system, the administrator must "fly blind." Too often, the only indicators the network manager has on the relative health of a particular server is whether or not it is running.

It is well-acknowledged that there is a lack of reliability and availability of most standards-based servers. Server downtime, resulting either from hardware or software faults or from regular maintenance, continues to be a significant problem with significant costs. With emerging Internet, intranet and collaborative applications taking on more essential business roles every day, the cost of network server downtime will continue to spiral upward.

While hardware fault tolerance is an important element of an overall high availability architecture, it is only one piece of the puzzle. Studies show that a significant percentage of network server downtime is caused by transient faults in the I/O subsystem. These faults may be due, for example, to the device driver, the device firmware or hardware, which does not properly handle concurrent errors, and often causes servers to crash or hang. The result is hours of downtime per failure while a system administrator discovers the failure, takes some action and manually reboots the server. In many cases, data volumes on hard disk drives become corrupt and must be repaired when the volume is mounted. A dismount-and-mount cycle may result from the lack of "hot pluggability" or "hot plug" in current standards-based servers. Hot plug refers to the addition and swapping of peripheral adapters to an operational computer system. An adapter is simply any peripheral printed circuit board containing microchips, such as a PCI card, that may be removed from or added to a server peripheral device slot. Diagnosing intermittent errors can be a frustrating and time-consuming process. For a system to deliver consistently high availability, it should be resilient to these types of faults.

Existing systems also do not have an interface to control the changing or addition of an adapter. Since any user on a network could be using a particular adapter on the server, system administrators need a software application that controls the flow of communications to an adapter before, during, and after a hot plug operation on an adapter.

Current operating systems do not by themselves provide the support users need to hot add and swap an adapter. System users need software that will freeze and resume the communications of their adapters in a controlled fashion. The software needs to support the hot add of various peripheral adapters such as mass storage and network adapters. Additionally, the software should support adapters that are designed for various bus systems such as Peripheral Component Interconnect, CardBus, Microchannel, Industrial Standard Architecture (ISA), and Extended ISA (EISA). System users also need software to support the hot add and swap of adapters within canisters, which are detachable bus casings for a detachable bus system, and which also provide multiple slots for adapters.

In a typical PC-based server, upon the failure of an adapter, the system must be powered down, the new adapter and adapter driver installed, the server powered back up and the operating system reconfigured. However, various entities have tried to implement the hot plug of these adapters to a fault tolerant computer system. One significant difficulty in designing a hot plug system is protecting the circuitry contained on the adapter from being short-circuited when an adapter is added to a powered system. Typically, an adapter contains edge connectors which are located on one side of the printed circuit board. These edge connectors allow power to transfer from the system bus to the adapter, as well as supplying data paths between the bus and the adapter. These edge connectors fit into a slot on the bus on the computer system. A traditional hardware solution for "hot plug" systems includes increasing the length of at least one ground contact of the adapter, so that the ground contact on the edge connector is the first connector to contact the bus on insertion of the I/O adapter and the last connector to contact the bus on removal of the adapter. An example of such a solution is described in U.S. Patent No. 5,210,855 to Bartol.

U.S. Patent No. 5,579,491 to Jeffries discloses an alternative solution to the hot installation of I/O adapters. Here, each hotly installable adapter is configured with a user actuable initiator to request the hot removal of an adapter. The I/O adapter is first physically connected to a bus on the computer system. Subsequent to such connection, a user toggles a switch on the I/O adapter which sends a signal to the bus controller. The signal indicates to the bus controller that the user has added an I/O adapter. The bus controller then alerts the user through a light emitting diode (LED) whether the adapter can be installed on the bus.

However, the invention disclosed in the Jeffries patent also contains several limitations. It requires the physical modification of the adapter to be hotly installed. Another limitation is that the Jeffries patent does not teach the hot addition of new adapter controllers or bus systems. Moreover, the Jeffries patent requires that before an I/O adapter is removed, another I/O adapter must either be free and spare or free and redundant. Therefore, if there was no free adapter, hot removal of an adapter is impossible until the user added another adapter to the computer system.

A related technology, not to be confused with hot plug systems, is Plug and Play defined by Microsoft® Corporation and PC product vendors. Plug and Play is an architecture that facilitates the integration of PC hardware adapters into systems. Plug and Play adapters are able to identify themselves to the computer system after the user installs the adapter on the bus. Plug and Play adapters are also able to identify the hardware resources that are needed for operation. Once this information is supplied to the operating system, the operating system can load the adapter drivers for the adapter that the user had added while the system was in a non-powered state. However, to date, Plug and Play has only been utilized for the hot docking of a portable computer to an expansion base.

Therefore, a need exists for improvements in server management which can result in continuous operation despite adapter failures. System users should be able to replace failed components, upgrade outdated components, and add new functionality, such as new network interfaces, disk interface adapters and storage, without impacting existing users. Additionally, system users need a process to hot add their legacy adapters, without purchasing new adapters that are specifically designed

for hot plug. As system demands grow, organizations must frequently expand, or scale, their computing infrastructure, adding new processing power, memory, mass storage and network adapters. With demand for 24-hour access to critical, server-based information resources, planned system downtime for system service or expansion has become unacceptable.

The improvements of co-pending applications entitled "Hot Add of Devices Software Architecture" and "Hot Swap of Devices Software Architecture," as well as their related applications, all filed on October 1, 1997, adds hot swap and hot add capabilities to server networks. The recent availability of hot swap and hot add capabilities requires that a user maintaining the server, usually a server network system administrator, knows or learns the numerous and complicated steps required to swap or add a peripheral device, including how to suspend the device adapters, how to power down and up the correct server slot and/or canister, etc. These steps are more fully disclosed in the co-pending applications referenced above and incorporated herein by reference. In addition, because servers have become very reliable, the system administrator will often be caught in a position of not knowing or having forgotten how to swap or add a peripheral adapter when a server malfunctions or when a new adapter needs to be added. Today's servers do not often malfunction, and the system administrator may add adapters only a few times a year.

Without detailed knowledge of the hot swap and hot add processes, the system administrator will be unable to change out and install peripheral devices. In that case, the entire server system must then be shut down, the peripheral adapter replaced or inserted, and the system restarted. This can result in severe losses to the system users in terms of network downtime and inability to service clients. In addition, it results in a failure to take advantage of the currently available hot swap and hot add technology. However, without an automated step-by-step process from the user's point of view, these results are inevitable.

Therefore, a need exists to automate, as much as possible, the hot swap and hot add processes, so that the benefits of those capabilities in a server are not compromised by insufficient technical knowledge of those processes on the part of the network administrator. Because implementation of the hot add and hot swap processes

only depends on (1) which process is necessary (i.e. hot swap or hot add) and (2) which particular server peripheral device slot is concerned, the user should be able to perform these processes knowing such information. The remaining steps in the hot swap and hot add processes may be completely automated. This can allow the necessary hot swapping and hot addition of adapters to be performed quickly and efficiently, by non-expert personnel, while the server is running. In the usual case, it would allow the system administrator to perform a hot swap or a hot add with little or no learning curve.

Summary of the Invention

The present invention provides a user of a typical client-server or similar system, usually a system administrator, with a simple, understandable user interface that allows the user to both view and transmit basic instructions to perform the steps in a hot plug or similar process in the system. In one embodiment of the invention, the system comprises a computer that includes memory, a module that operates to provide a user interface, and a module that transmits communications and instructions between that user interface and the computer to execute a hot plug process. The user interface may be a graphical user interface, and may comprise a series of screen displays capable of showing and executing the steps of a hot plug process for a peripheral adapter. The hot plug process may be a hot swap process, a hot add process, or other similar process.

Brief Description of the Drawings

Figure 1 is a block diagram showing a fault-tolerant computer system which uses one embodiment of the present invention.

Figure 2 is a diagram showing one embodiment of a server, having eight peripheral adapter slots.

Figure 3 is a diagram showing another embodiment of a server, having four canisters and sixteen peripheral adapter slots.

Figure 4 is a logic block diagram illustrating the software modules that may be used to implement the functions of one embodiment of the present invention.

Figure 5 is a block diagram illustrating the connection between a user interface and the peripheral adapters of a server which can be hot swapped or hot added, including other major parts of the system.

Figure 6 is a block diagram illustrating one embodiment of the present invention in a network server running the NetWare® operating system.

Figure 7 is a block diagram illustrating one embodiment of the present invention in a network server running the Windows® NT operating system.

Figure 8 is a flow diagram illustrating the steps of one embodiment of a hot swap process.

Figure 9 is a flow diagram illustrating the steps of one embodiment of a hot add process for a server having independent peripheral device slots (NF-9008) and running the NetWare® operating system.

Figure 10 is a flow diagram illustrating the steps of one embodiment of a hot add process for a server having peripheral device slot canisters (NF-9016) and running the NetWare® operating system.

Figures 11 through 18 show the screen displays generated by one embodiment of the graphical user interface for a hot swap process implemented in the NF-9008 server environment.

Figures 19 through 26 show the screen displays generated by one embodiment of the graphical user interface for a hot swap process implemented in the NF-9016 server environment.

Figures 27 through 32 show the screen displays generated by one embodiment of the graphical user interface for a hot add process implemented in the NF-9008 server environment running the NetWare® operating system.

Detailed Description of the Invention

The various features of the invention will be described with reference to a particular software module (referred to as the "Hot Plug PCI Wizard") and various related components. The Hot Plug PCI Wizard is part of the Maestro Central ("Maestro") computer program, sold by NetFRAME Systems, Inc. In these drawings, reference numbers are reused, where appropriate, to indicate a correspondence between referenced items. Moreover, although the following detailed description describes

particular embodiments of the invention, the invention can be embodied in a multitude of different ways as defined and covered by the claims. Finally, the following detailed description describes embodiments of the invention under the Windows® NT and the NetWare® operating systems. Alternative embodiments of the invention may use other commercial operating systems, such as MacIntosh® OS, OS/2, VMS, DOS, Windows® 3.1/95/98 or UNIX.

In addition, in the following description of the invention, a "module" includes, but is not limited to, software or hardware components which perform certain tasks. Thus, a module may include object-oriented software components, class components, procedures, subroutines, data structures, segments of program code, drivers, firmware, microcode, circuitry, data, tables, arrays, etc. A module also includes all components within the definition of "module" found in RFC 1213, *Management Information Base for Network Management of TCP/IP-based Internets: MIB-II*, which contains a module defining the basic objects needed to manage a TCP/IP network. Separate RFC documents contain modules defining objects for specific technologies such as Token-Ring interfaces, Open Shortest Path First (OSPF) routing, and Appletalk® networking. Those with ordinary skill in the art will also recognize that a module can be implemented using a wide variety of different software and hardware techniques.

Figure 1 presents an overview of a computer system in which the invention may be used. One or more servers 10 are used to support one or more clients 12 in a typical client-server network, which is made up of various hardware components, including standard microprocessors. The microprocessors used may be any conventional general purpose single- or multi-chip microprocessor such as a Pentium® processor, a Pentium® Pro processor, a 8051 processor, a MIPS® processor, a Power PC® processor, or an ALPHA® processor. In addition, the microprocessor(s) may be any conventional special purpose microprocessor such as a digital signal processor or a graphics processor. The microprocessor(s) will have conventional address lines, conventional data lines, and one or more conventional control lines.

A user at one of the client 12 monitoring stations uses the Simple Network Management Protocol (SNMP) Manager (Maestro) 14 to obtain information regarding the status of various machines and/or devices in the server, and to send instructions

to the server through an SNMP Agent 16 and various other system components as shown. The SNMP serves as a mechanism to provide and transport management information between network components, in order to manage the actual devices in a network. The SNMP was developed almost ten years ago as a standard for internetwork management. It has been widely published and is widely available today.

Maestro 14 at the client uses the SNMP to transmit instructions to the SNMP Agent 16 and SNMP Extension Agent 18, and vice versa. The SNMP permits interactive network administration via parameter checks and supervision of certain network conditions. SNMP uses a transport protocol stack such as the User Datagram Protocol/Internet Protocol (UDP/IP), Transmission Control Protocol/Internet Protocol (TCP/IP), DECnet (Digital Equipment Corporation network protocol), and others.

Maestro 14 provides the user, including network administrators, with a representation of the state of the target machine and/or devices in the server. It can provide this representation in simple (character-oriented) or graphical form. Maestro 14 herein described is graphical and has been specifically designed for the NF-9000 servers' Management Information Base, or MIB. A MIB is simply a virtual information database containing information necessary to manage devices in the network. The module types that represent management information transferred via the SNMP protocol through the network are gathered in the MIB 20. Thus, the MIB 20 contains variables that hold status information regarding the server(s) and other devices in the network. Maestro 14 accesses the data in the MIB to send proper instructions to and from the user over the network, using the SNMP. The devices mentioned in the discussion herein include peripheral cards or adapters that may be plugged into receiving slots in the server. The present invention also applies to any device that may be inserted into or removed from a server—that is, "hot pluggable" devices.

The SNMP instructions sent by Maestro 14 reach the SNMP Agent 16 and/or the SNMP Extension Agent 18 at the server end via the network drivers 26, communication hardware 28, and network medium. The SNMP Agent 16 and SNMP Extension Agent 18 wait for incoming requests and respond to them using information retrieved from the system services 22 such as device drivers, the Intrapulse 24

firmware, and other components of the operating system. These functions and the software and hardware involved are further described in the co-pending applications entitled "Hot Add of Devices Software Architecture" and "Hot Swap of Devices Software Architecture," and their related applications, filed October 1, 1997 and herein incorporated by reference.

The server architecture shown in Figure 2 represents the NetFRAME Model NF-9008 server. The NF-9008 supports eight (8) peripheral devices, or adapters, through eight (8) peripheral I/O slots 30. The slots 30 are individually powered and may be accessed directly from the PCI bus 32 and its associated bridge 34. The server architecture shown in Figure 3 represents the NetFRAME Model NF-9016 server. The NF-9016 supports sixteen (16) peripheral devices through sixteen (16) peripheral I/O slots 36. The NF-9016 additionally uses canisters 38, which are groups of four slots. In the NF-9016 each canister is accessed by the PCI Bus 40 and an associated bridge 42. Thus, it is necessary to extract a canister to change or add a device to one of the slots inside the canister; therefore, all the other boards in the canister get shut down. In the NF-9008 each peripheral device can be replaced independently of the others.

Figure 4 shows the explicit modular software implementation of one particular embodiment of the invention. The specific implementation shown and hereinafter described is implemented in the Microsoft® Windows® environment using the C++ programming language. Other software implementations of the invention may be developed by those of ordinary skill in the art.

Upon start-up of Maestro 14, it creates a module called EnumServer 44 which retrieves information regarding how many network servers are present in the system, and then it creates Server Modules 46 for each of those servers. These and all other "modules" described herein are simply computer software elements that can issue commands to execute functions, including calling other modules, and store data. The use of software modules is well understood in the relevant art.

Thus, because one module is created for each server in the network, the Server Modules 46 provide a physical, one-to-one representation of the server network hardware. The Server Modules 46 transmit and receive information and commands

to control their associated server through the MIB Manager Module 48, which uses the MIB 20 to translate the module variables and send commands to and from the SNMP Module 50, which can then send and receive logical numerical information over the network 52 to and from the SNMP Agents present at the server. The use of the MIB 20 is disclosed in detail in the copending application entitled "Data Management System Supporting Hot Plug Operations on a Computer," as well as its related applications, filed October 1, 1997 and hereby incorporated herein by reference.

The Windows® GUI Module 54 is a standard software module provided in the Microsoft® Windows® 95 and Windows® NT environments. This module implements the basic screen display format in the Microsoft® Windows® environment, which is used by one embodiment of the invention, NetFRAME's Hot Plug PCI Wizard, in creating its graphical user interface (GUI). The screen displays seen by the user upon the user's implementation of either the hot swap or hot add processes, shown herein in Figures 11 through 32, are customized GUI's implemented by the use of subsidiary Custom GUI Modules 56. For example, Custom GUI 1 Module 56 is used to create the screen display of Figure 11, and it contains the data shown in that screen display, including the text, graphics, and select buttons. All of the other customized screen displays are similarly created by an associated Custom GUI Module in the Microsoft® Windows® 95 and Windows® NT environments.

All of the GUI modules for the hot swap/hot add processes are associated with each other, using software pointers, in a manner that will enable them to directly call different screen displays when necessary to create an appropriate sequence of screen displays to implement the hot swap or hot add process. For example, referring to Figure 11, the "Back" button on the screen invokes a pointer to the previous custom screen module, the "Next" button invokes a pointer to the custom screen module representing the next step in the underlying process (i.e. hot add or hot swap), the "Cancel" button invokes a pointer to the initial custom screen module for the Hot Plug PCI Wizard, and the "Help" button invokes a pointer to a custom help screen module, which contains information to help answer the user's questions.

Referring again to Figure 4, when the Hot Plug PCI Wizard is first accessed by the user within Maestro 14, it creates the Custom GUI Modules 56 for the hot swap and hot add processes. At that time, Maestro 14 also creates modules 60, associated with the server modules 46, representing all of the server canisters, slots, and associated adapters present in the server network. These modules, like the server modules, are able to access and hold data regarding the status of their associated hardware at the server, and issue commands that may be used to manipulate that associated hardware. For example, the Custom GUI module shown in Figure 15, from which the user is able to power down a selected slot in a NF-9008 server, first calls the appropriate server Slot Module 60 (here the module for Slot 1 of a one-server NF-9008 network). The Slot Module 60 checks the status of its slotPowerState integer module variable to verify that the slot is powered up, and if so the Slot Module 60 can then issue a "PowerDown" command, through the MIB Manger Module 48 and SNMP Module 50 to the server, where the server hardware receives the instruction through an SNMP agent and executes the instruction to power down the slot. The particular slot is identified by data within the Slot Module 60, specifically the slotGroupNumber and slotNumber variables for this implementation of the invention. The values associated with these variables would be 1 and 1, respectively, for the first slot on the left, viewing the NF-9008 server from the front, 1 and 2 for the second slot from the left, and so on.

The appropriate canister and adapter modules 60 are called from the Custom GUI Modules 56, and commands issued therefrom, in an identical manner as will be understood by one of ordinary skill in the art. The specific module variables and data most often used in the hot swap/hot add processes are as follows for the particular modules:

MODULE	VARIABLES
Canister Module	canisterMaximumNumberOfCanisters (= 0 for NF-9008)
	canisterNumber
	canisterName
	canisterPowerState
Slot Module	slotGroupNumber

	slotNumber
	slotAdapterPresence
	slotPowerState
Adapter Module	adapterNumber
	adapterName
	adapterSupportsHotSwapHotAdd
	adapterState
	adapterCommand

The module variable names are self-explanatory. Thus, within the Maestro 14 software framework, the Hot Plug PCI Wizard implementation of the present invention is able to use Custom GUI Modules 56 for each screen display that the user sees when performing a hot swap or hot add process. These custom screen displays are easily implemented, and are linked to the server modules 46, canister, slot, and adapter modules 60, which are in one-to-one correspondence with the actual server hardware, and which modules (1) provide the user with status information regarding that hardware, and (2) allow the user to easily implement high-level software commands over the network to control the server hardware during the hot swap and hot add processes.

Figure 5 shows the basic hardware components at the server that respond to instructions generated by the Custom GUI Modules 56 generated by the Hot Plug PCI Wizard software in Maestro 14. In Figure 5, the specific instruction involved deals with hot swapping a peripheral adapter 62. As explained above, the server operating system could be Windows® NT or NetWare®, including others. First, the GUI 64 accepts a request by the user, such as a system manager or administrator, to perform a hot add or hot swap of a peripheral adapter 62 at the server. The GUI 64 transmits the user's instruction through the operating system 66 to the hot plug system driver 68 and the adapter driver 70 or drivers associated with the peripheral adapter 62. The hot plug system driver 68 controls the adapter driver 70 for a hot plug operation. The hot plug system driver 68 suspends and resumes the communications between the peripheral adapter 62 and the adapter driver 70. During the hot add or swap of the peripheral adapter 62, the hot plug hardware 72 deactivates the power to the peripheral

adapter, allowing the user to remove it from the server and replace it with another peripheral adapter. One embodiment of the hot plug hardware 72 may include a network of microcontrollers to carry out this functionality. The peripheral adapter 62 could be any kind of peripheral device, such as a math co-processor, a sound board, or other devices well known in the art.

Figure 6 is a block diagram illustrating the system components of the NetWare® implementation of one embodiment of the present invention. A configuration manager 74 is responsible for managing all of the peripheral adapters. The configuration manager 74 keeps track of the configuration information for every adapter. The configuration manager 74 also allocates resources for every adapter and initializes each adapter during a hot swap operation. The GUI 96 initiates the requests to the configuration manager 74 to freeze and restart communications to a specified peripheral adapter.

Novell® has created two interfaces for adapter drivers to communicate with the Netware Operating Systems. First, Novell® has provided the Open Datalink Interface (ODI) for network drivers. Second, Novell® has created the Netware Peripheral Architecture (NWPA) for mass storage adapters. Each of these interfaces will be briefly described.

With respect to network device drivers, such as a driver 76, ODI was created to allow multiple LAN adapters to co-exist on network systems. The ODI specification describes the set of interface and software modules used by hardware vendors to interface with the NetWare® operating system. At the core of the ODI is the link support layer (LSL) 78. The LSL 78 is the interface between drivers and protocol stacks (not shown). A protocol stack is a layered communication architecture, whereby each layer has a well defined interface.

Novell® has provided a set of support modules that provide the interface to the LSL 78. These modules are a collection of procedures, macros and structures. These modules are the media support module (MSM) 80, which contains general functions common to all drivers, and the topology specific modules (TSM) 82, which provide support for the standardized media types of token ring, Fiber Distributed Datalink Interface (FDDI) and Ethernet. The MSM 80 manages the details of interfacing ODI

multi-link interface drivers to the LSL 78 and the NetWare® Operating System. The MSM 80 handles all of the generic initialization and run-time issues common to all drivers. The topology specific module or TSM 82 manages operations that are unique to a specific media type. The Hardware Specific Modules (HSM) 84 are created by each adapter vendor for each type of peripheral adapter. The HSM 84 contains the functionality to initialize, reset and shutdown an adapter. The HSM 84 also handles packet transmission and reception to and from each adapter.

With respect to the mass storage device driver 86, the NetWare® Peripheral Architecture (NWPA) is a software architecture developed by Novell® which provides an interface for mass storage developers to interface with the NetWare® operating system. The NWPA is divided into two components: a host adapter module (HAM) 90 and a custom device module (CDM) 94. The HAM 90 is a component that contains information on the host adapter hardware. The CDM 94 is the component of the NWPA that regulates the mass storage adapters. The main purpose of the Filter CDM 94 is to locate each HAM 90, register for adapter events, and process the I/O suspend and I/O restart requests from the configuration manager 74.

Figure 7 is a block diagram illustrating various components of one embodiment of the present invention as implemented under the Windows® NT Operating System (WinNT). A configuration manager 100 controls the process of hot adding and swapping an adapter. An administrative agent 102 initiates requests to the configuration manager 100 and the network of microcontrollers to oversee the process of hot add and swap of an adapter. The administrative agent 102 initiates requests to the configuration manager 100 to suspend and restart the communications of a peripheral adapter. The administrative agent 102 initiates requests to the microcontroller network device driver 104 to turn on and off the power to the appropriate server slots.

The configuration manager 100 controls the communication between each adapter and adapter driver by calling the SCSI port 106 and NDIS 108. SCSI port and NDIS are interfaces which are exported by the Windows® NT Operating system and which are imported, respectively, into mass storage and network adapter drivers. These interfaces are designed to interact with a miniport 110

which is an instance of an adapter driver. In Windows® NT, each adapter (type of adapter if reentered) will have its own miniport.

The remaining Figures 8 through 32 delineate the hot swap and hot add processes as implemented by a user according to one embodiment of the invention, the Hot Plug PCI Wizard software. The Hot Plug PCI Wizard can operate in both the Windows® NT and NetWare® server environments, and the differences between those two implementations are noted herein.

Figure 8 shows the steps performed in the hot swap process under either the Windows® NT or the NetWare® operating environments. The custom GUI screen displays corresponding to each step in the hot swap process are shown in Figures 11 through 26. Before beginning the hot plug or hot add process, the user first accesses, through a network map window icon or menu, the server management window for the particular server of interest. The user can then enter the Hot Plug PCI Wizard to perform the hot swap or hot add process.

At the first screen, the user performs the first step, Select Operation, to select the operation they wish to perform—either "Hot Swap a PCI card" or "Hot Add a PCI card" (see Figures 11 and 19). In this case the user selects the former. Upon this selection the Custom GUI Module 56 for that screen retrieves data from the appropriate server module 46 to identify whether the server is a NF-9008, having 8 slots, or a NF-9016, having 4 canisters. The server module 46 obtained this information using the SNMP Module when the user started Maestro 14.

At the next screen the user selects the specific peripheral device slot where they wish to perform the hot swap operation. At this screen, the display varies depending on whether the server of interest is a Model NF-9008, having 8 independent slots, or NF-9016, having 4 canisters housing 4 slots each. The Custom GUI Module 56 identifies the server by retrieving that information from the server module 46 (see Figure 4), and then incorporates that knowledge to create either the customized screen display of Figure 12 or Figure 20. Thus, if the server is a NF-9008, the user will be displayed Figure 12, in which case the user will then select (i.e. single-click) on the slot of interest. If the server is a NF-9016, the user will be displayed Figure 20, in which case the user will select the canister first, and

then select the appropriate slot within that canister. As noted in Figure 4, in either case, the user will only be allowed to select slots that contain adapters, because only those support the hot swapping process. The Custom GUI Module 56 accesses the adapter modules 60 (see Figure 4) to determine which ones (if any) are not hot swappable. This information is contained in the adapter module variable called "adapterSupportsHotSwapHotAdd."

Again referring to Figure 8, the third step in the hot swap process is simply a confirmation screen, which allows the user to see the steps that the software will instruct the server hardware to perform, and prompts the user to continue when ready (see Figures 13 and 21). Next, the hot swap process begins at the server with the suspension of the requested slot adapters, upon the user activating the "Suspend Adapter(s)" button on the screen (see Figures 14 and 22). Upon receiving this instruction, the Custom GUI Module for that screen prompts the specified adapter module(s) to send an adapterCommand, "SuspendOperations," to the MIB Manager Module, which then transfers this message to the SNMP Module. The SNMP Module can then send this SuspendOperations command over the network via the SNMP, to the SNMP Agent. The SNMP Agent then transmits the command to the server hardware, and the appropriate hardware devices are engaged to suspend the affected adapter(s).

Once the adapters are suspended, the SNMP Agent transmits that status information back over the network to the SNMP Module and back through the MIB Manager Module, to the adapter object. The state of the adapterState variable in the adapter module is updated to "Suspended," and the next Custom GUI module is in communication with this module to recognize that the appropriate adapters have been suspended and the GUI can then move on to the next step in the process. As noted in Figure 8, for a server running in the NetWare® environment, only one step is required to both suspend the appropriate adapters and power down the appropriate slot or canister. Thus, the next step for a NetWare® implementation is to actually swap out the intended card.

The next step for a Windows® NT implementation is to power down the appropriate slot, if the server is a NF-9008, or the appropriate canister, if the server

is a NF-9016. In the NF-9016, individual slots may not be powered down; instead, the power must be suspended to the entire canister. The screen displays for the power down steps are shown in Figures 15 and 23. Once again, the Custom GUI Module 56 for this step receives the user's confirmation to go ahead with this step, and the module then proceeds to access the slotPowerState or canisterPowerState variable information from the slot or canister module 60, transmit a "PowerDown" command to the MIB Manager Module 48 and through the network via the SNMP Module 50. Once the appropriate slot or canister is powered down, the slotPowerState or canisterPowerState receives that information and updates the slot or canister module 60, and the Custom GUI Module recognized from that information that it may proceed to the next step in the hot swap process.

In either the NetWare® or Windows® NT environment, the next step in the hot swap process is to prompt the user to replace the peripheral card in the selected slot. This allows the user to physically go to the server, find the appropriate slot, and swap out and replace the peripheral card. The screen display for this step, shown in Figures 16 and 24, includes instructions to the user to make sure the LED light at the selected slot is off, and to make sure that the new card is correctly inserted into the slot. Again, once the user has finished this step, the screen display prompts the user to double-click on the "Next" button to proceed to the next step in the hot swap process. For the Windows® NT environment, the nest step is to power back up the affected slot or canister (see Figures 17 and 25), which the Custom GUI Module 56 for that screen display accomplishes by instructing the slot or canister module 60 to issue a "PowerUp" command to the MIB Manager Module 48 and through the network via the SNMP Module 50. Once the appropriate slot or canister is powered back up, the slotPowerState or canisterPowerState receives that information and updates the slot or canister module 60, and the Custom GUI Module recognized from that information that it may proceed to the next, and final, step. As stated above, in the NetWare® environment, this step is not necessary.

The final step in the hot swap process is to restart the adapters that were previously suspended before swapping out the peripheral card. This step is

performed by the user from the last screen display, shown in Figures 18 and 26, by the user activating the "Restart Adapter(s)" command. The "Restart Adapter(s)" command prompts the adapter module 60 to issue a "ResumeOperations" command to the MIB Manager Module 48 and through the network via the SNMP Module 50. Once the user receives confirmation through the adapter module that the state of the adapters (i.e. adapterState) is "Active," the swapped peripheral card may be used in the server.

Figures 9 and 10 show the hot add process steps for the NF-9008 (Figure 9) and NF-9016 (Figure 10) servers. The process steps are shown for the NetWare® environment, although the process may also be implemented in the Windows® NT environment as will be understood by one of ordinary skill in the art. The hot add process is very similar to the hot swap process, except that in a hot add process, a peripheral card is inserted in a server slot that previously did not hold a card.

The screen displays for the hot add process for a NF-9008 server are shown in Figures 27 through 32. Figure 27 shows that the first step in the hot add process is the same screen display as for the hot swap process, except that this time the user selects "Hot Add a PCI card." When the user selects the hot add option, the Custom GUI Module verifies from the state of the server object that the server is a NF-9008, and then verifies from the adapter and/or slot modules 60 that there is no peripheral card presently in that particular server slot. Once confirmation is received by the Custom GUI Module, in the next step, from the screen display shown in Figure 28 the user selects the particular slot in which the peripheral card will be added, in a manner similar to the identical step in the hot swap process for the NF-9008 server, described above. The next screen display, shown in Figure 29, is simply a confirmation screen, similar to Figure 13 for the hot swap process, which just allows the user to hit the "Next" button to prompt the next screen. The next step in the hot add process is for the user to go to the server and physically insert the new peripheral card in the slot at the server. The screen display for this step is shown in Figure 20. After hitting the "Next" button on the screen display, the user is prompted at the next screen display, shown in Figure 31, to hit "Power Up" when ready to power the server slot up with the newly inserted peripheral

card. At this step, the Custom GUI Module 54 accesses the appropriate slot module 60 to send a PowerUp command via the MIB Manager Module 48, the SNMP Module 50, and the network to the physical server slot, where the command operates to power up the previously inactive slot with the added adapter. Finally, the last user screen in the hot add process instructs the user to make sure to configure the new peripheral card for use. These steps to configure the new card are listed as shown in Figure 31, and the configuration process for newly added peripheral cards is readily understood by those of ordinary skill in the art.

For the NF-9016 hot add operation, because the NF-9016 contains its peripheral card slots within canisters, the entire canister must be powered down and then back up in order to add a new card to one of the slots within a canister. For this reason the hot add process for the NF-9016 is nearly identical to the hot swap process for the NF-9016. The process steps in the hot add process are the identical steps implemented in the hot swap process (see Figures 8 and 19-26); the only difference is that when the user goes out to the peripheral card slot, the user only has to insert the new peripheral card rather than first removing an already resident card.

Thus, with respect to each of hot swap and hot add processes herein described and carried out by Maestro 14, the user is able to successfully complete a hot swap or a hot add of an adapter from the user's computer workstation screen, and is able to do so knowing no information about these processes other than which process is needed, and which particular server peripheral device slot is concerned.